

(19) KOREAN INTELLECTUAL PROPERTY OFFICE

## KOREAN PATENT ABSTRACTS

(11)Publication number: 1020020072969 A

(43)Date of publication of application: 19.09.2002

(21)Application number: 1020010013070

(71)Applicant: CHUNGNAM NATIONAL UNIVERSITY

(22)Date of filing: 14.03.2001

(72)Inventor: LEE, GYU CHEOL

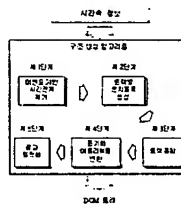
(51)Int. Cl. G06F 17/24

(54) SMIL EDITOR AND EDITING METHOD THEREOF

(57) Abstract:

PURPOSE: A SMIL(Synchronized Multimedia Integration Language) editor and an editing method thereof are provided to automatically generate the SMIL structure for a multi-media presentation based on a predetermined logic.

CONSTITUTION: An object defining unit defines the object of the SMIL as a standard language for realizing a multi-media presentation on the web. A function defining unit defines the relation between the objects defined by the object defining unit. A structure generating unit automatically generates the structure of the presentation by analyzing the structure information of the presentation to realize through the object defining unit and the function defining unit from the time base information between the objects. To edit the SMIL, the relation between objects forming a presentation is defined. Thereafter, the structure information of the presentation is analyzed. Upon automatic generation of the structure of the presentation, the presentation is realized.



&amp;copy; KIPO 2003

## Legal Status

Date of request for an examination (20010314)

Notification date of refusal decision (00000000)

Final disposal of an application (abandonment)

Date of final disposal of an application (20040508)

Date of registration (00000000)

Date of opposition against the grant of a patent (00000000)

BEST AVAILABLE COPY

(19) 대한민국특허청(KR)  
(12) 공개특허공보(A)

(51) Int. Cl.<sup>7</sup>  
G06F 17/24

(11) 공개번호 특2002-0072969  
(43) 공개일자 2002년09월19일

(21) 출원번호 10-2001-0013070  
(22) 출원일자 2001년03월14일  
(71) 출원인 대한민국(충남대학교)  
대전 유성구 궁동 220번지  
(72) 발명자 이규철  
대전광역시서구삼천동991국화아파트203동108호  
(74) 대리인 박병철

심사청구 : 있음

(54) 멀티미디어 동기화 언어 편집기 및 그 편집방법

요약

본 발명은 복잡하고, 전문화된 SMIL 문서를 제작하기 위해서 사용자 중심의 편리한 인터페이스를 제공하는 SMIL 편집기 및 그 편집방법에 관한 것으로써, 특히 SMIL 지향적인 구조 편집이 아닌 사용자 지향적인 시간축 중심의 편집 윈도우를 제공하며, 상기 시간축 정보로부터 자동으로 SMIL 구조정보가 파악되고 그 구조가 생성되도록 하여 SMIL 문법에 대한 구체적인 지식이 없는 사용자로 하여금 잘 정의된 SMIL 문서를 제작하는 것을 가능하게 하는 동시에 프레젠테이션을 구성하는 SMIL 문서의 전체 시나리오의 수정이 용이하도록 할 수 있는 효과가 있다.

대표도

도5

색인어

SMIL, 미디어 객체, 병렬 블록, 순차 블록, 구조정보, 시간축 정보, 멀티미디어, 동기화, XML

명세서

도면의 간단한 설명

도 1은 종래 SMIL 편집기의 제1 예를 도시한 도,  
도 2는 종래 SMIL 편집기의 제2 예를 도시한 도,  
도 3은 종래 SMIL 편집기를 사용한 편집과정의 문제점을 도시한 도,  
도 4는 본 발명에 따른 SMIL 편집기에서 편집하고자 하는 객체들을 시간축을 기준으로 도시한 도,  
도 5는 본 발명에 따른 SMIL 편집방법에서 이루어지는 구조생성과정의 알고리즘이 도시된 도,  
도 6은 도 5의 제1 단계에 따른 제1 실시예의 처리결과가 도,  
도 7은 도 5의 제1 단계에 따른 제2 실시예의 처리결과가 도,  
도 8은 도 5의 제1 단계에 따른 도 4의 처리결과가 도시된 도,  
도 9는 도 5의 제2 단계의 처리결과가 도시된 도,  
도 10은 도 5의 제3 단계의 처리결과가 도시된 도,  
도 11은 도 5의 제4 단계의 처리결과가 도시된 도,  
도 12는 도 5의 제5 단계의 처리결과가 도시된 도이다.

<도면의 주요부분에 대한 설명>

A-H : 미디어 객체

## 발명의 상세한 설명

### 발명의 목적

#### 발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 멀티미디어 동기화 언어 편집기 및 그 편집방법에 관한 것으로서, 특히 사용자가 구현하고자 하는 프레젠테이션의 구조정보를 파악하고 그에 따라 자동적으로 상기 프레젠테이션의 구조가 생성되도록 하는 멀티미디어 동기화 언어 편집기 및 그 편집방법에 관한 것이다.

멀티미디어 동기화 언어(Synchronized Multimedia Integration Language; 이하 SMIL라 칭함)는 웹 상에서 텔레비전과 유사한 형태의 멀티미디어 프레젠테이션(Presentation)을 쉽게 작성할 수 있도록 하기 위한 W3C(World Wide Web Consortium)의 표준언어이다. 그러나 XML(Extensible Markup Language)기반의 SMIL 문서는 구조적인 문서 구조를 가지기 때문에 사용자가 단순한 텍스트 편집기를 사용하여 SMIL을 이용한 멀티미디어 프레젠테이션을 구현하는 것은 쉽지 않은 일이다.

이 때문에, 다양한 형태의 GUI(Graphic User Interface)를 사용한 SMIL 문서 저작 도구, 즉 SMIL 편집기가 발표되었다.

Oratrix사에서 제공하는 SMIL 편집기 중 하나인 GRINS(A Graphic Interface for Creating and Playing SMIL Documents)는 윈도우나 매킨토시, 유닉스 기반의 다양한 환경을 지원하는 SMIL 기반의 멀티미디어 프레젠테이션 편집기이다. GRINS는 구조 편집기와 타임라인 뷰(View)를 제공하여 SMIL의 시간적 동기화 정보를 기술할 수 있도록 하고 있다.

도 1은 GRINS 편집기를 사용하여 SMIL 문서를 편집한 예를 보여준다. GRINS의 구조 편집기는 사용자가 <seg>, <par> 엘리먼트를 상자모양의 인터페이스를 사용하여 편집하도록 한다. 사용자는 상기 GRINS 구조 편집기를 사용하여 구현하고자 하는 프레젠테이션의 개략적인 전체 시나리오를 구성하고, 타임라인 뷰를 통하여 이를 확인할 수 있다.

Sausage Software사에서 개발한 SMIL Composer Supertoolz는 자체 플레이어는 제공하지 않으며, Real Network사의 Real Player G2를 대상으로 한다.

상기 SMIL Composer Supertoolz는 상기 GRINS와는 달리 타임라인에 대한 사용자 인터페이스를 제공하지 않는다. 따라서, 직관적으로 프레젠테이션을 이루는 미디어 객체 구현 시간을 확인하기 힘들다. 즉, 사용자는 각 객체 구현 시간을 확인하기 위하여 직접 시간 계산을 해야 한다.

도 2는 SMIL 문서를 상기 SMIL Composer Supertoolz를 사용하여 편집한 화면이다.

그러나 상기와 같은 종래의 SMIL 편집기는 SMIL 문서가 가지는 문서구조를 사용자가 직접 편집하여 전체 시나리오를 구성해야 한다. 따라서, 사용자는 SMIL에 대한 사전지식이 필요하여 무엇보다도 시나리오의 편집에 많은 어려움이 따른다.

도 3은 시나리오 편집 시 나타나는 구조 편집의 문제점을 도시하고 있다. 도 3의 (a)는 도 3의 (b)에서 미디어 객체가 점선으로 표시된 수정부분이 반영되기 이전의 시나리오를 위한 SMIL 문서이다. 도 3의 (b)에 점선으로 표시된 수정부분을 반영하기 위해서 편집기를 사용할 경우에 사용자는 전체적인 구조를 허물고 도 3의 (c)이나 (d)와 같은 형태로 전체 문서를 재구성해야 한다. 즉, 시나리오의 미세한 수정을 위하여 시나리오의 구조 전체가 바뀌어져야 한다는 문제점이 있다.

또한, 상기의 문제점에서 보여지듯 하나의 시나리오가 다양한 형태의 SMIL 문서로 만들어 질 수 있으며, 원하는 시나리오를 얻기 위한 구조정보를 파악하고 구조를 생성하기 위하여 많은 노력이 필요하다는 문제점이 있다.

#### 발명이 이루고자 하는 기술적 과제

본 발명은 상기한 종래 기술의 문제점을 해결하기 위하여 안출된 것으로서, 그 목적은 사용자에게 보다 간편한 형태의 인터페이스를 제공할 수 있도록 시간축을 기준으로 사용자가 구현하고자 하는 프레젠테이션의 시나리오를 작성하여 상기 프레젠테이션을 이루는 SMIL 구조를 정해진 로직에 따라 자동적으로 생성할 수 있도록 하는 멀티미디어 동기화 언어 편집기 및 그 편집방법을 제공하는데 있다.

### 발명의 구성 및 작용

상기한 과제를 해결하기 위한 본 발명에 의한 멀티미디어 동기화 언어 편집기의 특징에 따르면, 웹 상에서 멀티미디어 프레젠테이션을 구현할 수 있도록 하는 표준언어인 멀티미디어 동기화 언어(SMIL)의 객체를 정의하는 객체정의수단과, 상기 객체정의수단으로 정의된 객체사이의 관계를 정의하는 함수정의수단과, 사용자가 상기 객체정의수단과 함수정의수단을 통해 구현하고자 하는 프레젠테이션의 구조정보를 상기 함수정의수단에 의해 관계가 정의된 객체사이의 시간축 정보로부터 파악하여 상기 프레젠테이션의 구조를 자동적으로 생성하는 구조 생성수단으로 구성된다.

또한, 본 발명에 의한 멀티미디어 동기화 언어 편집방법의 제1 특징에 따르면, 사용자가 구현하고자 하는 프레젠테이션을 이루는 객체 및 상기 객체사이의 관계를 정의하는 제1 단계와, 상기 제1 단계에서 정의된

객체사이의 시간축 정보로부터 구현하고자 하는 프레젠테이션의 구조정보를 파악하는 제2 단계와, 상기 제2 단계에서 구조정보가 파악된 프레젠테이션의 구조가 자동적으로 생성되고 그에 따라 상기 프레젠테이션이 구현되는 제 3단계로 이루어진다.

또한, 본 발명에 따른 멀티미디어 동기화 언어 편집방법의 제2 특징에 따르면, 구현하고자 하는 프레젠테이션을 이루는 객체사이의 시간축 정보를 파악하는 제1 단계와, 상기 제1 단계에서 파악된 정보에 따라 상기 객체사이의 인과관계를 파악하고 상기 인과관계에 따라 상기 객체들을 계층적으로 정리하는 제2 단계와, 상기 제2 단계에서 계층적으로 정리된 객체들이 이루는 복수개의 트랙을 순차적인 블록으로 변환하는 제3 단계와, 상기 제3 단계에서 변환된 순차 블록을 하나의 병렬 블록으로 변환하는 제4 단계와, 상기 제4 단계에서 병렬 블록을 이루는 각 객체들이 구현되는 시작시간 및 종료시간에 따라 상기 객체들을 동기화하는 제5 단계와, 상기 제5 단계에서 동기화된 객체들을 통해 구현하고자 하는 프레젠테이션의 구조가 생성되고 그에 따라 상기 프레젠테이션이 구현되는 제6 단계로 이루어진다.

그 외에, 본 발명에 의한 멀티미디어 동기화 언어 상에서 인과관계를 가지는 객체의 편집방법의 특징에 따르면, 제1 트랙의 제1 객체와 인과관계를 가지는 제2 트랙의 제2 객체가 상기 제1 객체와 병렬 구조를 가지도록 계층적으로 정리되는 제1 단계와, 상기 제1 단계에서 인과관계를 가지는 상이한 트랙의 객체들이 계층적으로 정리됨에 따라 상기 정리된 객체들이 하나의 객체로 인식되는 제2 단계로 이루어진다.

마지막으로, 본 발명에 의한 멀티미디어 동기화 언어로 이루어진 객체를 동기화하는 편집방법의 특징에 따르면, 프레젠테이션을 이루는 객체들 중 제1 객체가 구현되는 동기화 시작시간이 상기 제1 객체와 병렬 구조를 통해 다른 객체와 병렬 블록을 이루었을 경우 상기 제1 객체와 상기 병렬 블록의 구현 시작시간 차이가 되고, 상기 제1 객체가 순차 구조를 통해 다른 객체와 순차 블록을 이루었을 경우 상기 제1 객체의 구현 시작시간과 상기 제1 객체 전에 구현되는 객체의 구현 종료시간의 차이가 되는 제1 단계와, 상기 제1 단계에서 동기화 시작시간이 정의된 제1 객체의 구현이 지속되는 동기화 지속시간이 상기 제1 객체가 병렬 구조를 통해 다른 객체와 병렬 블록을 이루었을 경우 가장 늦게 종료되는 상기 병렬블록의 구현 종료시간과 상기 병렬 블록의 구현 시작시간 차이가 되고, 상기 제1 객체가 순차 구조를 통해 다른 객체와 순차 블록을 이루었을 경우 마지막 객체의 구현 종료시간과 상기 순차 블록의 구현 시작시간 차이가 되는 제2 단계로 이루어진다.

이하, 본 발명의 바람직한 실시예를 첨부된 도면을 참조하여 상세히 설명한다.

본 발명에 의한 SMIL 편집기는 시간축을 기준으로 구현하고자 하는 프레젠테이션을 이루는 미디어 객체들의 구현 시작시간 및 지속시간에 관한 정보를 포함하여 저장하는 자료구조를 가지는 객체정의수단과 함수정의수단, 그리고 사용자가 상기 객체정의수단과 함수정의수단을 통해 구현하고자 하는 프레젠테이션의 구조정보를 상기 함수정의수단에 의해 관계가 정의된 객체사이의 시간축 정보로부터 파악하여 상기 프레젠테이션의 구조를 자동적으로 생성하는 구조 생성수단으로 구성된다.

여기서, 본 발명에 따른 SMIL 편집기는 SMIL 문법 중심의 구조 편집을 지양하고 사용자 중심의 직관적인 시간축 편집기를 사용하여 사용자가 프레젠테이션을 위한 전체 시나리오를 편집할 수 있도록 하면서 결과로서 구조적인 모델을 가지는 SMIL 문서를 생성한다.

그리고, 상기 시간축 정보는 사용자가 구현하고자 하는 프레젠테이션의 시나리오를 구성하기 위해 사용되는 시간축 편집 윈도우에 대한 내부 데이터 모델이기도 하다. 아래의 표 1은 시간축 정보가 저장되는 객체들에 대해 정의한 것이다.

[표 1]

객체	정의
MO	미디어 객체(Media Object) animation, audio, image, video, text
SO	동기화 객체(Synchronization Object) seq, par
SB	동기화 블록(Synchronization Block) 하나 이상의 MO와 SO로 이루어진 트리 (1)MO는 SB이다. (2)자식노드 가지는 모든 노드는 SO이다. (3)모든 리프 노드들은 MO이다. (4)형제노드 간의 순서관계가 존재한다.
T	트랙(Track) 크기가 n인 SB들의 배열로써 (1) $n \geq 0$ (2)if $1 \leq j < n$

상기 표 1에서 동기화 블록 SB는 동기화 객체 <seq>, <par> 와 같은 엘리먼트 및 상기 엘리먼트에 포함된 미디어 객체인 노드들로 이루어진 하나의 블록을 의미한다. 그리고, 사용자 편집 시 상기 트랙 T는 특정

한 영역에 구현되는 미디어 객체의 집합을 의미하지만 시간의 중복 없이 순차적으로 표현되는 동기화 블록 SB의 집합으로 정의할 수도 있다.

다음의 표 2는 상기 객체의 관계를 정의하는 함수는 정의한 것이다.

[표 2]

<i>SB next(SB SBij)</i>
SBij가 j번째 트랙의 j번째 원소라 할 때, <i>next(SBij)</i> 는 <i>SBi(j+1)</i>
<i>Time start(SB SBij)</i>
SBij의 구현 시작시간
<i>Time end(SB SBij)</i>
SBij의 구현 종료시간
<i>SB constructSeqBlock(SB SBij, Time t)</i>
SetSB={SBij} ∪ {SBix   SBij, SBix ∈ T i and <i>end(SBij) ≤ start(SBix) &lt; t</i> 일 때, root가 'seq'인 SDO이고, SetSB에 포함된 모든 SB들을 자식노드로 갖는 SB
<i>Bool isEventSource(SB SBa, SB SBg)</i>
SBa가 SBg의 event source 관계에 있으면 true, 그렇지 않으면 false

도 4는 시간축을 중심으로 구현하고자 하는 객체를 도시한 것이다. 여기서 'A-H'는 미디어 객체 MO인 동시에 동기화 블록 SBO이며, 점선으로 그려진 둥근 사각형들은 시간축을 중심으로 구현하고자 하는 프레젠테이션을 이루는 트랙들은 나타낸다. 동기화 블록 B에 대한 *next(B)*는 동기화 블록 C를 의미하며, *start(B)*와 *end(B)*는 각각 동기화 블록 B의 구현 시작과 종료시간인 30과 60을 의미한다.

도 4에 도시된 바와 같이, 미디어 객체사이의 이벤트 기반의 시간관계는 미디어 객체 B와 E사이, 그리고 F와 H 사이에만 존재하며, 따라서 상기 객체 B와 E 그리고 F와 H에만 인과관계가 존재한다. 그러므로 *isEventSource(E,B)*와 *isEventSource(F,H)*는 true 값을 돌려준다. 또한, *constructSeqBlock(F,80)*은 동기화 블록 F와 G를 자식노드로 하는 새로운 순차 블록의 생성을 의미한다.

상기한 도 4에 도시된 미디어 객체들과 같이 2차원 배열 형태를 가지는 객체들로부터 SMIL 문서를 생성하기 위해서는 시간축 정보로부터 구조정보를 파악하고 그에 따라 구조를 생성하기 위한 구조 생성과정이 필요하다.

상기 구조 생성과정이 이루어지도록 동작하는 구조 생성수단의 구조생성 알고리즘은 상기 각 객체의 시간축 정보를 파악함으로써 사용자의 표현 및 구현 의도를 유지하면서 저작 결과를 SMIL 문서로 표현하기 위해 사용된다. 이를 통해 사용자는 SMIL에 대한 상세한 구문 규칙과 같은 사전지식 없이 웹 상에서 교란이 용이하며, 표준화된 방법에 의해 기술되어진 SMIL 문서를 저작하고 편집할 수 있다.

도 5는 상기 구조 생성수단에서 이루어지는 구조 생성과정을 도시한 것으로서, 시간축 정보가 다섯 단계의 처리과정을 거쳐 최종 결과물로서 구현하고자 하는 프레젠테이션의 구조인 DOM(Document Object Model) 트리로 생성되어짐을 보여준다.

먼저, 제1 단계에서 상기 구조 생성수단은 서로 다른 트랙에 존재하며 서로 독립적으로 구현되는 상기 객체들이 가지는 이벤트 기반의 시간관계 즉, 인과관계를 설정하여 상기 트랙 사이의 명시적인 관련성을 모두 제거한다. 따라서, 인과관계에 있는 미디어 객체들은 병렬 표현을 의미하는 <par> 블록으로 변환되어 병렬 구조를 가지는 병렬 블록을 이루게 된다.

여기서, 상기 구조 생성수단은 상기 이벤트 기반의 시간관계를 상기 인과관계를 가지는 미디어 객체들의 관계에 따라 다음의 두 가지 경우로 나누어 처리함으로써 각 미디어 객체들 간의 병렬성을 최대한 보장하면서 사용자의 의도가 보다 정확하게 반영되도록 한다.

도 6에 도시된 제1 실시예의 경우 동기화 블록 'A'가 이벤트의 기준으로서 동기화 블록 'C'와 이벤트 기반의 시간관계에 있을 때 동기화 블록 'C'로부터 시작하는 순차 블록 중 동기화 블록 'A'의 다음에 표현되는 동기화 블록 'B'의 구현 시작시간 이전에 종료되는 것이 존재하는 경우이다.

도 6의 (a)는 상기 제1 실시예를 정의하고 있으며, (b)는 상기 (a)의 정의에 따른 제1 실시예의 시간관계를 도시하고, (c)는 그에 따른 구조 생성수단의 처리결과를 도시한다.

도 6의 (c)에 도시된 바와 같이, 동기화 블록 'A'와 'C'에 대한 이벤트 기반의 시간 관계는 이벤트의 기준인 동기화 블록 'A'와 함수 *constructSeqBlock(C,t)*에 의해 생성된 새로운 동기화 블록을 자식노드로 가지는 병렬 동기화 블록으로 변환된다.

도 7은 이벤트 기반의 시간관계에 대한 제2 실시예를 도시하였다. 상기 제1 실시예와 달리 제2 실시예에서는 동기화 블록 'D'가 *next(A)*의 시작시간 t에 걸쳐 구현되는 것을 알 수 있다. 이러한 경우 제1 실시예와 같은 생성과정을 거치면 동기화 블록 'B'가 변환된 동기화 블록사이에 시간의 겹침 현상이 발생함으로써 상기 표 1에 정의된 트랙 T의 정의에 위배된다.

따라서, 도 7의 (c)에 도시된 바와 같이 *constructSeqBlock(A,t1)*와 *constructSeqBlock(C,t)*의 의해 생성

된 새로운 순차 동기화 블록들로 구성된 병렬 동기화 블록으로 변환된다.  
 도 8은 도 4의 시간축 정보에 따른 상기 제1 단계의 처리결과를 보여준다.

다음의 표 3은 constructSeqBlock 함수에 대한 변환 알고리즘을 표 4는 상기 제1 단계에서 이루어지는 과정에 대한 처리 알고리즘을 기술하고 있다. 여기서 상기 제1 단계 과정은 전체 트랙을 우로부터 좌로 순회하면서 이벤트 기준이 되는 미디어를 만났을 때 수행한다. 제1 단계 변환 과정에 대한 처리 함수인 doFirstStep은 처리과정에서 constructSeqBlock 함수를 호출하게 되며 constructSeqBlock 함수에서 다시 doFirstStep 함수를 호출하는 순환구조를 가지고 있다.

[표 3]

```

SBref constructSeqBlock( SB SBij, Time t ) {
    Assumption:
    SBij 는 j번째 트랙의 j번째 SB
    SBx 는 임의의 SB
    SetSB = SBij ∪ { SBix | x > j }
    Procedure:
    For SBk ∈ SetSB
        doFirstStep( SBk )

    SetSBseq = { SBix | x ≥ j and
    start( SBix ) ≤ t }
    If( N( SetSBseq ) == 1 )
        SBref = SBix
    Else {
        SBref = create new SB
        SBref.root = SO 'seq'
        SBref.children = SetSBseq
    }
    return SBref
}
    
```

[표 4]

```

doFirstStep( SB SBij ) {
Assumption:
트랙 Ti 는 SBij를 포함하는 i번째 트랙
SetSBevents = { SBx | isEventSource( SBij, SBx ) = true
}
Procedure:
For SBx ∈ SetSBevents {
t = start( next( SBij ) );
SBevents = constructSeqBlock( SBx, t )
If( end( SBevents ) ≤ t )
    /// case-I
    SBbase = SBij
Else {
    /// case-II
    SBbase = create new SB
    SBbase.root = SO 'seq'
    SBbase.children = SBij U next( SBij )
}
    SBpar = create new SB
    SBpar.root = SO 'par'
    SBpar.children = SBbase U SBevents
}
}
    
```

제2 단계에서, 상기 구조정보 생성수단은 상기 제1 단계에서 사용자가 명시한 모든 객체 사이의 이벤트 기반의 시간관계가 하나의 객체로서 인식될 수 있는 병렬 동기화 블록으로 변환됨에 따라 트랙간의 명시적인 인과관계가 제거되었으므로 각 트랙에 남아있는 동기화 블록들이 순차적인 관계를 가지면서 다른 트랙에 독립적으로 표현되도록 한다. 즉, 도 9에 도시된 바와 같이, 상기 제1 단계 후 각 트랙에 남아있는 동기화 블록들은 하나의 순차 블록으로 변환한다.

제3 단계에서, 상기 구조정보 생성수단은 상기 제2 단계 후 각 트랙이 공집합이거나 하나의 순차 동기화 블록을 가짐에 따라 각 트랙이 다른 트랙에 독립적으로 동시에 표현되어지므로 상기 각 트랙의 순차 동기화 블록들이 하나의 병렬 동기화 블록으로 통합되도록 한다. 이에 따라 도 9에 도시된 바와 같은 형태의 트리로 이루어진 구조가 생성되며, 이 트리는 XML 문서를 생성하기 위한 가장 기본적인 형태의 트리이다.

여기서, 상기 시간축에 포함된 미디어 객체의 구현 시작시간 및 지속시간은 시간축 상에서 '0'으로부터 시작하는 절대적인 시간이거나 이벤트 소스를 기준으로 하는 시간임으로 <seq>, <par> 엘리먼트의 중첩관계와 각 미디어 객체의 동기화 어트리뷰트(attribute)를 사용하여 미디어 객체의 구현 시간을 명시하는 XML 과는 상이한 구조를 가지고 있다.

따라서, 제4 단계에서 상기 구조정보 생성수단은 상기 시간축의 시간이 변환되어질 미디어 객체의 동기화 어트리뷰트(attribute)로 변환되도록 한다. 여기서 동기화 어트리뷰트에는 동기화 시작시간 지정을 위한 'begin' 과 지속시간 지정을 위한 'dur'이 사용된다.

다음 표 5와 표 6은 각각 <par>, <seq> 블록의 자식노드들에 대한 동기화 시작시간의 값을 구하는 알고리즘을 보여준다. 동기화 시작시간의 값은 시간값(clock value)과 이벤트 값(element-event value)으로 구분되어진다. 따라서, 시간축에서 이벤트 기반의 시간관계에 있는 미디어 객체의 동기화 어트리뷰트의 값은 이벤트 값으로 변환되어야 한다.

공통적으로 이벤트 기반의 시간관계에 있는 미디어 객체들의 동기화 시작시간은 'id(source\_id)( event )'와 같은 형태의 이벤트 값으로 변환되어진다. 그 외의 일반적인 미디어 객체들은 <par> 블록에서는 자신과 <par> 블록의 시작 시간과의 차를, <seq> 블록에서는 자신과 앞선 형제노드의 종료시간과의 차를 동기화 시작시간의 값으로 한다.

또한, 각 미디어 객체의 동기화 지속시간은 편집시에 사용자에게 의해 명시되어지며, <par> 블록의 지속시간은 '가장 늦게 종료되는 자식노드의 종료 시간과 블록의 시작시간의 차'로 설명할 수 있으며, <seq> 블록의 지속시간은 '마지막 자식노드의 종료시간과 블록의 시작시간의 차'로 설명할 수 있다.

표 5와 표 6은 상기에 따른 동기화 시작시간 및 지속시간에 대한 알고리즘을 각각 기술하고 있다.

[표 5]

```

setBeginValue(SB SBi)
Assumption:
    SBi의 최상위 노드가 'par'인 S00이고,
    SBi의 자식노드 집합을 SetSBoh라 할 때,

Procedure:
    For SBj ∈ SetSBoh {
        If(SBj has any event source ? )
            SBj.begin = 'id(source_id)(event)'
        Else
            SBj.begin=
            toString(start(SBj)-start(SBi))

        setBeginValue( SBj )
    }
    
```

[표 6]

```

setBeginValue(SB SBi)
Assumption:
    SBi의 최상위 노드가 'seq' S00이고,
    SBi의 자식노드 집합을 SetSBoh라 할 때,

Procedure:
    SBtemp = SBi
    For SBj ∈ SetSBoh {
        If(SBj has any event source ? )
            SBj.begin = 'id(source_id)(event)'
        Else
            SBj.begin=start(SBj)-end(SBtemp)

        setBeginValue( SBj )
        SBtemp = SBj
    }
    
```

도 11은 동기화 어트리뷰트를 변환한 결과를 예를 들어 보여주고 있다. 도 11의 (a)는 시간축 정보에 포함된 각 미디어 객체의 구현 시작시간과 지속시간을 절대 시간으로 표현한 것이며, 도 11의 (b)는 동기화 시작시간과 지속시간으로 변경한 것이다.

상기 제4 단계에서 변환되어진 결과는 병렬 블록을 최상위 노드로 하는 하나의 트리 형태를 가진다. 이러한 결과 트리는 구조 생성 알고리즘의 단순화와 프로그램 작성의 편의를 위하여 필요하지 않은 병렬 노드를 포함하게 된다.

제5 단계는 상기 구조 생성수단의 구조 생성 알고리즘의 최종단계로써 이 단계에서 상기 구조 생성수단은 최소한의 엘리먼트를 가지는 SMIL 문서를 작성하기 위한 최적화 단계를 수행하며, XML문서로서의 적법한 SMIL 문서를 생성하기 위해 최적화된 트리로부터 도 11에 도시된 바와 같이 DOM 트리를 생성한다.

여기서, 도 11의 (a)는 시간축을 중심으로 하는 각 미디어 객체들이 배열된 것을 도시하였으며, 도 11의 (b)는 중첩된 <per> 엘리먼트를 가지는 병렬 동기화 블록을 도시하고, 도 11의 (c)는 상기 도 11의 (b)에



서 중첩된 <per> 엘리먼트를 제거한 트리를 도시하고 있다. OOM 트리의 각 노드는 제 5 단계에 의해 최적화된 구조 생성 결과 트리의 노드와 1:1로 대응되어진다.

#### 발명의 효과

상기와 같이 구성되는 본 발명의 SMIL 편집기 및 그 편집방법은 SMIL 지향적인 구조 편집이 아닌 사용자 지향적인 시간축 중심의 편집 윈도우를 제공하며, 상기 시간축 정보로부터 자동으로 SMIL 구조정보가 파악되고 그 구조가 생성되도록 하여 SMIL 문법에 대한 구체적인 지식이 없는 사용자로 하여금 잘 정의된 SMIL 문서를 저작하는 것을 가능하게 하는 동시에 프레젠테이션을 구성하는 SMIL 문서의 전체 시나리오의 수정이 용이하도록 할 수 있는 효과가 있다.

#### (57) 청구의 범위

**청구항 1.** 웹 상에서 멀티미디어 프레젠테이션을 구현할 수 있도록 하는 표준언어인 멀티미디어 동기화 언어(SMIL)의 객체를 정의하는 객체정의수단과; 상기 객체정의수단으로 정의된 객체사이의 관계를 정의하는 함수정의수단과; 사용자가 상기 객체정의수단과 함수정의수단을 통해 구현하고자 하는 프레젠테이션의 구조정보를 상기 함수정의수단에 의해 관계가 정의된 객체사이의 시간축 정보로부터 파악하여 상기 프레젠테이션의 구조를 자동적으로 생성하는 구조 생성수단을 포함하여 구성된 것을 특징으로 하는 멀티미디어 동기화 언어 편집기.

**청구항 2.** 제 1 항에 있어서,

상기 객체정의수단과 상기 함수정의수단은 사용자가 이를 활용하여 요구하는 프레젠테이션을 구현할 수 있도록 상기 사용자에게 인터페이스를 제공하는 인터페이스 수단을 포함하여 구성된 것을 특징으로 하는 멀티미디어 동기화 언어 편집기.

**청구항 3.** 제 1 항에 있어서,

상기 객체정의수단과 상기 함수정의수단은 각각 상기 객체 및 상기 관계가 정의된 객체의 구현 시작시간 및 지속시간에 관한 정보를 시간축을 기준으로 저장하고 있는 자료구조를 가지도록 구성된 것을 특징으로 하는 멀티미디어 동기화 언어 편집기.

**청구항 4.** 제 1 항에 있어서,

상기 구조 생성수단은 상기 프레젠테이션을 이루는 각 객체가 구현되는 시작시간 및 종료시간과 상기 객체사이의 인과관계에 따라 상기 프레젠테이션의 구조정보를 파악하고 그에 따라 상기 프레젠테이션의 구조를 계층적으로 생성하여 상기 프레젠테이션이 구현되도록 구성된 것을 특징으로 하는 멀티미디어 동기화 언어 편집기.

**청구항 5.** 웹 상에서 멀티미디어 프레젠테이션을 구현할 수 있도록 하는 표준언어인 멀티미디어 동기화 언어(SMIL)를 편집하는 방법에 있어서,

사용자가 구현하고자 하는 프레젠테이션을 이루는 객체 및 상기 객체사이의 관계를 정의하는 제1 단계와; 상기 제1 단계에서 정의된 객체사이의 시간축 정보로부터 구현하고자 하는 프레젠테이션의 구조정보를 파악하는 제2 단계와; 상기 제2 단계에서 구조정보가 파악된 프레젠테이션의 구조가 자동적으로 생성되고 그에 따라 상기 프레젠테이션이 구현되는 제 3단계를 포함하여 이루어진 것을 특징으로 하는 멀티미디어 동기화 언어 편집방법.

**청구항 6.** 제 5 항에 있어서,

상기 제2 단계는 상기 프레젠테이션을 이루는 각 객체가 구현되는 시작시간 및 종료시간과 상기 객체사이의 인과관계에 따라 상기 프레젠테이션의 구조정보를 파악하는 것을 특징으로 하는 멀티미디어 동기화 언어 편집방법.

**청구항 7.** 제 5 항에 있어서,

상기 제3 단계는 상기 제2 단계에서 파악된 구조정보에 따라 상기 프레젠테이션의 구조를 계층적으로 생성하여 상기 프레젠테이션이 구현되도록 하는 것을 특징으로 하는 멀티미디어 동기화 언어 편집방법.

**청구항 8.** 웹 상에서 멀티미디어 프레젠테이션을 구현할 수 있도록 하는 표준언어인 멀티미디어 동기화 언어(SMIL)를 편집하는 방법에 있어서,

구현하고자 하는 프레젠테이션을 이루는 객체사이의 시간축 정보를 파악하는 제1 단계와; 상기 제1 단계에서 파악된 정보에 따라 상기 객체사이의 인과관계를 파악하고 상기 인과관계에 따라 상기 객체들을 계층적으로 정리하는 제2 단계와; 상기 제2 단계에서 계층적으로 정리된 객체들이 이루는 복수개의 트랙을

순차적인 블록으로 변환하는 제3 단계와; 상기 제3 단계에서 변환된 순차 블록을 하나의 병렬 블록으로 변환하는 제4 단계와; 상기 제4 단계에서 병렬 블록을 이루는 각 객체들이 구현되는 시작시간 및 종료시간에 따라 상기 객체들을 동기화하는 제5 단계와; 상기 제5 단계에서 동기화된 객체들을 통해 구현하고자 하는 프레젠테이션의 구조가 생성되고 그에 따라 상기 프레젠테이션이 구현되는 제6 단계를 포함하여 이루어진 것을 특징으로 하는 멀티미디어 동기화 언어 편집방법.

**청구항 9.** 제 8 항에 있어서,

상기 제5 단계는 상기 동기화된 객체 중 중첩된 병렬 구조를 가지는 각 객체들을 하나의 병렬 구조로 통합하여 구현하고자 하는 프레젠테이션의 구조를 최적화하는 최적화단계를 더 포함하여 이루어진 것을 특징으로 하는 멀티미디어 동기화 언어 편집방법.

**청구항 10.** 웹 상에서 멀티미디어 프레젠테이션을 구현할 수 있도록 하는 표준언어인 멀티미디어 동기화 언어(SMIL)를 편집하는 방법에 있어서,

제1 트랙의 제1 객체와 인과관계를 가지는 제2 트랙의 제2 객체가 상기 제1 객체와 병렬 구조를 가지도록 계층적으로 정리되는 제1 단계와; 상기 제1 단계에서 인과관계를 가지는 상미한 트랙의 객체들이 계층적으로 정리됨에 따라 상기 정리된 객체들이 하나의 객체로 인식되는 제2 단계를 포함하여 이루어진 것을 특징으로 하는 멀티미디어 동기화 언어 상에서 인과관계를 가지는 객체의 편집방법.

**청구항 11.** 웹 상에서 멀티미디어 프레젠테이션을 구현할 수 있도록 하는 표준언어인 멀티미디어 동기화 언어(SMIL)를 편집하는 방법에 있어서,

프레젠테이션을 이루는 객체들 중 제1 객체가 구현되는 동기화 시작시간이 상기 제1 객체가 병렬 구조를 통해 다른 객체와 병렬 블록을 이루었을 경우 상기 제1 객체와 상기 병렬 블록의 구현 시작시간 차이가 되고, 상기 제1 객체가 순차 구조를 통해 다른 객체와 순차 블록을 이루었을 경우 상기 제1 객체의 구현 시작시간과 상기 제1 객체 전에 구현되는 객체의 구현 종료시간의 차이가 되는 제1 단계와; 상기 제1 단계에서 동기화 시작시간이 정의된 제1 객체의 구현이 지속되는 동기화 지속시간이 상기 제1 객체가 병렬 구조를 통해 다른 객체와 병렬 블록을 이루었을 경우 가장 늦게 종료되는 상기 병렬블록의 구현 종료시간과 상기 병렬 블록의 구현 시작시간 차이가 되고, 상기 제1 객체가 순차 구조를 통해 다른 객체와 순차 블록을 이루었을 경우 마지막 객체의 구현 종료시간과 상기 순차 블록의 구현 시작시간 차이가 되는 제2 단계를 포함하여 이루어진 것을 특징으로 하는 멀티미디어 동기화 언어로 이루어진 객체를 동기화하는 편집방법.

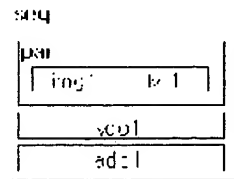
**청구항 12.** 제 11 항에 있어서,

상기 제1 단계는 상기 제1 객체가 상기 제1 객체의 구현이 시작되도록 하는 제2 객체가 있어 상기 제1 객체와 상기 제2 객체 사이에 인과관계가 있을 경우, 상기 제1 객체의 동기화 시작시간을 상기 제2 객체의 동기화 시작시간을 기준으로 정의하는 것을 특징으로 하는 멀티미디어 동기화 언어로 이루어진 객체를 동기화하는 편집방법.

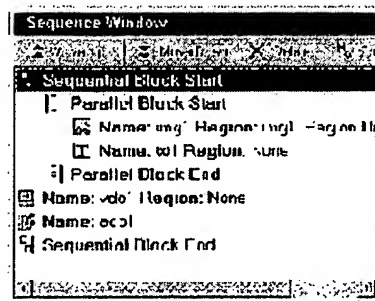
**도면**

**도면1**

```
<seq>
  <par>
    <img id="img1" dur="3" seq-on="n1"/>
    <act id="act1" region="r1"/>
  </par>
  <video id="vid1" dur="2" region="r1"/>
  <audio id="adcl"/>
</seq>
```



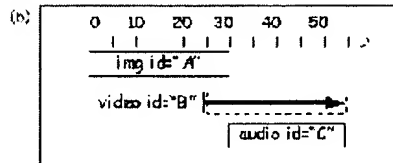
예 D2



예 D3

(a) 

```
seq
  par
    img id="A" dur="30"/>
    video id="B" dur="25"/>
  /par
  audio id="C" dur="25"/>
/seq
```



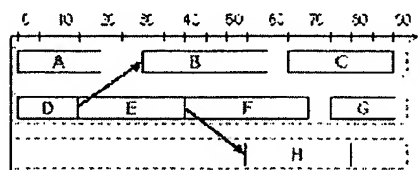
(c) 

```
par
  img id="A" dur="30"/>
  video id="B" dur="25"/>
  audio id="C" dur="25"
    begin="30"/>
/par
```

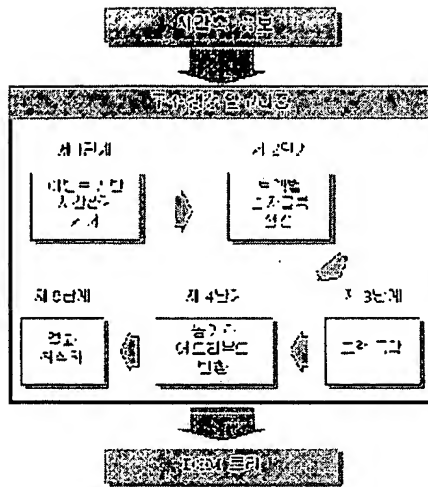
(d) 

```
par
  seq
    img id="A" dur="30"/>
    audio id="C" dur="25"/>
  /seq
  video id="B" dur="55"/>
/par
```

예 D4

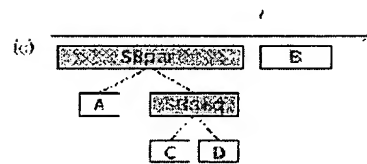
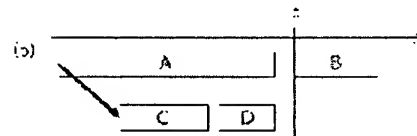


도 15



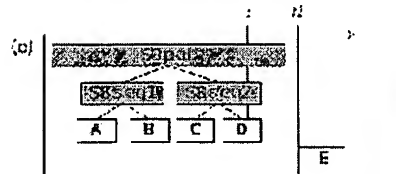
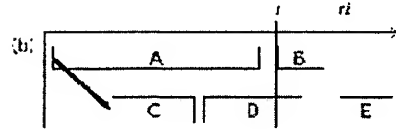
도 16

- (a) (1) isEventSource(c)의 값이 true  
 (2) endConstructSeqBlock(i-1) <= startConstructSeqBlock(i)  
 where i = start(i, next(i))

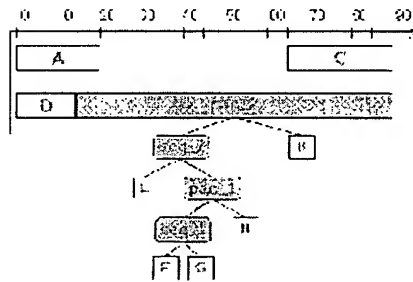


도 27

(a) (1) isEventSource(A, C) = NO true  
 (2) endConstructingBlock(C, i) > 1  
 return f\_start, next(A)



도 28



도 29

